AD-A034 666 HARRY DIAMOND LABS ADELPHI MD F/G 12/1
A PROGRAM FOR THE NUMERICAL INVERSION OF THE LAPLACE TRANSFORM.(U)
AUG 75 A HAUSNER
UNCLASSIFIED HDL-TR-1707 NL

1 OF 1
AD-A
034 666

END
DATE
FILMED
3-7-77
NTIS

A PROGRAM FOR THE NUMERICAL INVERSION OF THE LAPLACE TRANSFORM

HARRY DIAMOND LABORATORIES, ADELPHI, MARYLAND

AUGUST 1975

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER HDL-TR-1707	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Program for the Numerical Inversion of the Laplace Transform		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Arthur Hausner		8. CONTRACT OR GRANT NUMBER(s) DA: 1T161102A33B
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Prog El: 6.11.02.A Work Unit: 061A9
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Materiel Development & Readiness Command Alexandria, VA 22333		12. REPORT DATE August 1975 13. NUMBER OF PAGES 47
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES HDL Project No.: 302431 DRCMS Code: 611102.11.71200		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Laplace transform Inverse		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A program is given that numerically inverts a given Laplace transform for discrete times specified by the user, and (most of the time) to within a specified absolute error. The computation is adaptive in that the program decides whether to compute a given point by integrating the Bromwich integral		

Con'td

$$F(t) = \frac{1}{2\pi i} \int_{\gamma-i\infty}^{\gamma+i\infty} e^{st} f(s)\, ds, \quad t \neq 0$$

or to interpolate the point from the set of those already computed by integrating the Bromwich integral. If the integration is performed, it is done by accelerating partial sums to the limit, with the partial sums obtained by Gaussian quadrature with error control. Three examples, including two in which the transform is not the quotient of polynomials, indicate that the reliability of the program is good.

CONTENTS

1. INTRODUCTION

The Laplace transform is used frequently in engineering analysis primarily because many problems are easily formulated with it. The final solution requires inverting a tranform into the time domain, sometimes not possible in terms of common functions. Hence, many numerical inversion techniques have evolved. Most of the techniques[1-4] use the basic definition

$$f(s) = \int_0^\infty e^{-st} F(t)\, dt, \; t>0 \; , \tag{1}$$

evaluate f(s) for selected real s_i and specific t_j, and set up a matrix equation for solution, perhaps after a transformation to allow the use of orthogonal functions to simplify the solution. While these methods are extremely fast when done on a computer, they are limited in the accuracy obtainable and the class of f(s) to which they can be applied. Functions of time that are discontinuous or "nonsmooth" in some manner cannot be readily obtained. Berger and Duangudom also point out[5] that Berger's method[3] may even suffer from accuracy loss both for large t and smooth oscillatory functions such as $F(t) = \sin t$.

The techniques referenced above are not applicable to Laplace transforms obtained in the analysis of certain fluid transmission lines, where they are of the form

$$f(s) = \frac{1}{s \cosh\,[G(s)]} \; . \tag{2}$$

The inverse of equation (2) may be discontinuous [depending on the nature of G(s)] because of wave reflections in the line. Although these problems can also be formulated for solution by the method of

---

[1] *C. Lanczos, Applied Analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ (1956)pp. 284-303.*

[2] A. Papoulis, *"A New Method of Inversion of the Laplace Transform," Quarterly of Applied Mathematics,* 14 *(1957).*

[3] *B. S. Berger, "Inversion of the N-Dimensional Laplace Transform," Mathematics of Computation,* 20 *(1966), pp. 418-421.*

[4] *R. Bellman, R. E. Kalaba, and J. A. Lockett, Numerical Inversion of the Laplace Transform, American Elsevier Publishing Co., New York (1966).*

[5] *B. S. Berger and S. Duangudom, "A Technique for Increasing the Accuracy of the Numerical Inversion of the Laplace Transform with Applications," ASME Journal of Applied Mechanics, paper No. 73-WA/APM-1 (December 1973).*

Preceding page blank

characteristics and other methods, it was felt that a general-purpose inversion program for arbitrary Laplace transforms would have excellent utility.

The method developed is similar to that of Schmittroth[6] except that an error control is provided. It is based on integration of the Bromwich integral formula[7]

$$F(t) = \frac{1}{2\pi i} \int_{\gamma - i\infty}^{\gamma + i\infty} e^{st} f(s)\, ds, \quad t \neq 0 \ , \qquad (3)$$

where $s = x + iy$ (x,y real) and the integration in the complex plane is performed along the line $x = \gamma$. The number γ is arbitrary but must be chosen so that the line $x = \gamma$ lies to the right of all singularities (poles, branch points, or essential singularities).

This method is slower than other methods because $F(t)$ is obtained point by point by computing an infinite integral. To help speed matters along, the final algorithm contains an adaptive scheme that decides whether or not to interpolate a given point from the set of those then existing. Despite the relative slowness, good results have been obtained because of the current high speeds of digital computers. Ease of use and general utility of the program are its prime assets.

2. TRANSFORMATIONS TO A REAL INTEGRAL

With

$$s = \gamma + iy \qquad (4)$$

$$ds = i\, dy \qquad (5)$$

equation (3) transforms to

$$F(t) = \frac{e^{\gamma t}}{2\pi} \int_{-\infty}^{\infty} e^{iyt} f(\gamma + iy)\, dy \ , \qquad (6)$$

[6] *L. A. Schmittroth, "Numerical Inversion of Laplace Transforms," ACM Communications (March 1960), pp. 171-179.*

[7] *F. Scheid, Theory and Problems of Numerical Analysis, McGraw-Hill Book Co., Shaum's Outline Series, New York (1968), p. 125 ff.*

or

$$F(t) = \frac{e^{\gamma t}}{2\pi} \int_{-\infty}^{0} (\cos y't + i \sin y't)\, f(\gamma+iy')\, dy'$$

$$+ \frac{e^{\gamma t}}{2\pi} \int_{0}^{\infty} (\cos yt + i \sin yt)\, f(\gamma+iy)\, dy \; . \tag{7}$$

With the transformation y = -y' in the first integral, (7) reduces to

$$F(t) = \frac{e^{\gamma t}}{2\pi} \int_{0}^{\infty} \{(\cos yt + i \sin yt)\, f(\gamma+iy)$$

$$+ (\cos yt - i \sin yt)\, f(\gamma-iy)\}\, dy \; . \tag{8}$$

We now make use of conjugate properties of complex numbers

$$a^*b^* = (ab)^*$$

$$a^* + b^* = (a + b)^* \; . \tag{9}$$

If f(s) is expressed by a Taylor series (we assume analyticity)

$$f(s) = \sum_{j=0}^{\infty} a_j s^j \; , \tag{10}$$

we have

$$[f(s)]^* = \left[\sum_{j=0}^{\infty} a_j s^j\right]^* = \sum_{j=0}^{\infty} \left(a_j s^j\right)^*$$

$$= \sum_{j=0}^{\infty} a_j^* \left(s^j\right)^* = \sum_{j=0}^{\infty} a_j^* (s^*)^j \; . \tag{11}$$

Further assuming the a_j real,[8] then

$$[f(s)]^* = \sum_{j=0}^{\infty} a_j(s^*)^j = f(s^*) \ . \qquad (12)$$

Writing

$$f(\gamma+iy) = \text{Re}(f) + i\ \text{Im}(f) \qquad (13)$$

we have from equation (12)

$$f(\gamma-iy) = \text{Re}(f) - i\ \text{Im}(f) \ . \qquad (14)$$

Equation (8) reduces to

$$F(t) = \frac{e^{\gamma t}}{\pi} \int_0^{\infty} \{\text{Re}[f(\gamma+iy)] \cos yt - \text{Im}[f(\gamma+iy)] \sin yt\}\, dy \ . \qquad (15)$$

Since $F(-t) = 0$, the component parts of equation (15) are equal, but of opposite sign. Thus

$$\begin{aligned} F(t) &= \frac{2e^{\gamma t}}{\pi} \int_0^{\infty} \text{Re}[f(\gamma+iy)] \cos yt\, dy \\ &= -\frac{2e^{\gamma t}}{\pi} \int_0^{\infty} \text{Im}[f(\gamma+iy)] \sin yt\, dy \ . \end{aligned} \qquad (16)$$

We use the first integral in equation (16) as the basic one to be evaluated. With the final transformation

$$\begin{aligned} \omega &= yt \\ d\omega &= t\, dy \ , \end{aligned} \qquad (17)$$

[8] *R. V. Churchill, Complex Variables and Applications, 2nd ed., McGraw-Hill Book Co., New York (1960). (The result, eq (12), is the so-called "Principle of Reflection." The simplest test to determine if a_j is real is that f(s) is real whenever s is real.)*

our transformed integral becomes

$$F(t) = \frac{2e^{\gamma t}}{\pi t} \int_0^\infty \mathrm{Re}\left[f\left(\gamma + \frac{i\omega}{t}\right)\right] \cos \omega \, d\omega \ . \qquad (18)$$

3. COMPUTING THE TRANSFORMED INTEGRAL

Subroutine TPOINT computes the transformed integral [eq (18)] (see the listing in appendix A for a description of the arguments). To do this, the infinite integral is first changed to an infinite sum of finite integrals. Arbitrarily taking one cycle of the cos ω factor as the range of integration for each finite integral, we define

$$F_j(t) = \frac{1}{\pi} \int_{2\pi(j-1)}^{2\pi j} \mathrm{Re}\left[f\left(\gamma + \frac{i\omega}{t}\right)\right] \cos \omega \, d\omega \qquad (19)$$

so that

$$F(t) = \frac{2e^{\gamma t}}{t} \sum_{j=1}^{\infty} F_j(t) \ . \qquad (20)$$

Two problems remain in computing $F(t)$. First, each $F_j(t)$ must be computed accurately. Second, the infinite sum in equation (20) must be changed to a finite sum, say over N terms. To minimize N, we also intend to apply some nonlinear transformation algorithm to the sequence X_m of partial sums,

$$X_m = \sum_{j=1}^{m} F_j(t) \qquad (21)$$

in order to accelerate it to the limit as $m \to \infty$.

3.1 <u>Computing $F_j(t)$</u>

After some experimentation with QATR, the IBM 360 scientific routine[9] to perform quadrature integration, it was decided that the routine was inadequate to the task of computing $F_j(t)$. To obtain fast

[9]*"System/360 Scientific Subroutine Package, Version III, Programmer's Manual," Application Program GH20-0205-4 (1968)*, pp. 297-298.

convergence in computing equation (20), it is desirable to compute with γ close to the singularities. This, in turn, produces large peaks in the integrand of equation (19) so that QATR frequently returned with the error code indicating that the accuracy could not be reached because of rounding errors. In that case the smallest distance of γ from the singularities was doubled and the procedure repeated. The procedure turned out too time-consuming even for the simplest transforms.

The following procedure was adopted to compute equation (19) for each j:

(1) Compute equation (19) with an m_k-point Gaussian quadrature formula, where m_k is either 6, 12, 16, 24, 32, 40, 48, 64, 80, or 96, for k = 1, 2, ..., 10, arbitrarily using $m_k = 32$ for j = 1. Compute again, using an m_{k+1} formula. If the two results are G_k and G_{k+1}, we require that

$$\left|G_k - G_{k+1}\right| \leq E/10 , \tag{22}$$

where E is an absolute error parameter supplied by the user.

(2) If equation (22) is satisfied, $F_j(t)$ is taken to be G_{k+1}. If not, the subscript of m is increased until equation (22) is satisfied.

(3) The first pair of subscripts for which equation (22) is satisfied is also used in the next interval (j increased). If in addition

$$\left|G_k - G_{k+1}\right| \leq E/1000 , \tag{23}$$

the subscript k is decreased by one for the initial try in the next interval (unless k = 1).

(4) If k = 9 and equation (22) is not satisfied, the distance that γ is from a singularity is doubled and step 1 is repeated.

(5) If γ t > 11, the procedure is aborted. (This has never happened.)

To compute equation (19) by a Gaussian quadrature formula, we first let

$$\omega = \pi(z+1) + 2\pi(j-1) \tag{24}$$

to transform the limits of z from $[2\pi(j-1),\ 2\pi j]$ to $[-1,1]$. Thus, equation (19) transforms to

$$\begin{aligned} P_j(t) &= \int_{-1}^{1} \mathrm{Re}\left[f\left(\gamma + \frac{i(\pi z+2j-1)}{t}\right)\right] \cos\left[\pi(z+2j-1)\right] dz \\ &= -\int_{-1}^{1} \mathrm{Re}\left[f\left(\gamma + \frac{i(\pi z+2\pi j-\pi)}{t}\right)\right] \cos \pi z \, dz \, , \end{aligned} \tag{25}$$

since $2j-1$ is an odd integer. An m_k Gaussian quadrature formula computes[7]

$$I = \int_{-1}^{1} g(z)\, dz \approx \sum_{n=1}^{m_k} W_n' \, g(z_n) \, , \tag{26}$$

where the abscissas z_n and the weights W_n' are well tabulated for a variety of m_k. It is convenient to store the data

$$P_n = \pi z_n \tag{27}$$

to eliminate this multiplication during the course of the solution. Similarly, it is convenient to store

$$W_n = -\left(\cos P_n\right) W_n' \tag{28}$$

since these are also independent of j. From equations (25) to (28), the Gaussian quadrature formula used is then

---

[7] *F. Scheid, Theory and Problems of Numerical Analysis, McGraw-Hill Book Co., Shaum's Outline Series, New York (1968), p. 125 ff.*

$$F_j(t) = \sum_{n=1}^{m_k} W_n \operatorname{Re}\left[f\left(\gamma + \frac{i\{P_n + \pi(2j-1)\}}{t}\right)\right]. \qquad (29)$$

The algorithm described is very efficient, once the correct k is found in the first interval, because successive intervals are usually similar to one another, and there is never any need to compute the cosine function.

Condition (22) does not imply that F_j is computed to within an error E, but experience has shown that is the case for most of the examples tried. Errors, however, can accumulate in X_m. The final error in F(t) is subject to these errors and the errors due to the acceleration methods described below.

3.2 Accelerating the Sequence X_m to the Limit

The sequence X_m in equation (21) obtained from summing $F_j(t)$, $j = 1, 2, \ldots, m$, approaches the limit F(t) as $m \to \infty$, if γ is to the right of all singularities. This is the result of the Bromwich integral theorem. Since convergence may be very slow it is prudent to consider some acceleration technique. Subroutine TEAS[9] seemed ideal for this purpose. This subroutine, however, was frequently fooled by the sequences obtained from transforms of the type given in equation (2). After some experimentation with the Shanks algorithm,[10] upon which TEAS is based, it was decided that the algorithm was too sensitive to truncation errors, since each $X_m - X_{m-1}$ could have an absolute error of about E. It was therefore decided to simply apply an e_1 transformation[10] (i.e., an Aitken δ^2 transformation) to a modified subsequence of X_m, as described below.

[9] *"System/360 Scientific Subroutine Package, Version III, Programmer's Manual," Application Program GH20-0205-4 (1968), pp. 234-237.*

[10] *D. Shanks, "Non-Linear Transformation of Divergent and Slowly Convergent Sequences,"* 34 *(1955), pp. 1-42.*

Aitken's δ^2 transformation generates a sequence Q_m, $m = 3, 4, \ldots, N$, from a sequence X_m, $m = 1, 2, \ldots, N$, with

$$Q_m = X_m - \frac{\left(X_m - X_{m-1}\right)^2}{X_m - 2X_{m-1} + X_{m-2}} \; . \tag{30}$$

If $\lim_{m\to\infty} X_m = L$, and $L-X_m$ approaches zero nearly geometrically, then Q_m approaches the limit L faster[10] than X_m. This nonlinear transformation is frequently used to accelerate infinite sums and can be extremely effective to this end. The subsequence to be chosen is aimed at obtaining differences that are nearly geometric.

Figure 1 shows three types of sequences obtained by the procedure described. Type (a) eventually converges monotonically to a limit with the absolute difference between successive values becoming smaller. Type (b) is similar to type (a) except that the limit is approached via oscillation. Type (c) is a combination of (a) and (b), but the oscillation is such that the maximums get larger or the minimums get smaller.

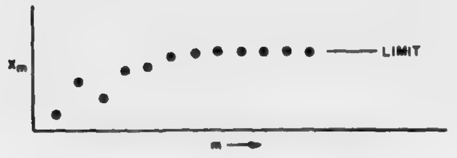
More complicated oscillatory sequences are obtained with some Laplace transforms, and it is difficult to devise an algorithm that will never be fooled. For example, figure 2 shows a sample sequence actually obtained for equation (2) where $G(s) = \sqrt{s+s^2}$. After a small number of intervals, it appears to be a type (b) sequence, but actually the high-frequency oscillation is superimposed on a low-frequency oscillation, and the limit is considerably greater than the apparent one.

In order to minimize the false limit possibility, the following procedure is used:

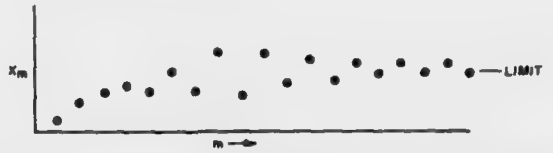
(1) A minimum of 11 points is used, i.e., the integration is carried to at least 22π.

(2) A type (a) sequence is assumed if 11 consecutive points are monotonic with successive absolute differences becoming smaller.

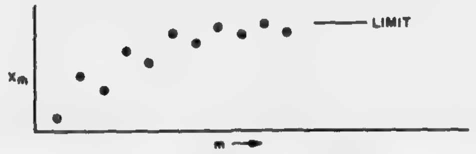
[10] *D. Shanks, "Non-Linear Transformation of Divergent and Slowly Convergent Sequences,"* <u>*34*</u> *(1955), pp. 1-42.*



(a) Eventually approaches a limit monotonically (successive differences get smaller).



(b) Eventually approaches a limit via oscillation where the envelopes of the maximums get smaller and the minimums get larger.



(c) Eventually approaches a limit via oscillation where the envelopes of the maximums get larger or the minimums get smaller.

Figure 1. Typical sequences X_m.

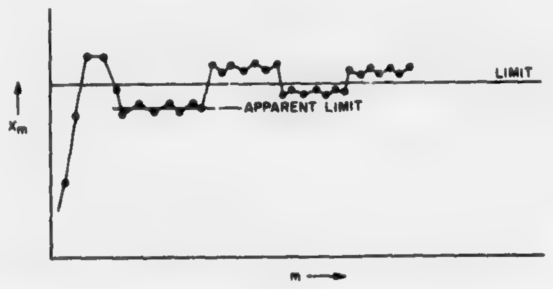


Figure 2. Example of a complicated sequence.

(3) Peaks (extremums) are stored in a peak array P. A peak is assumed if $(X_i - X_{i-1})(X_{i-1} - X_{i-2}) < 0$ and a parabola is passed through the three points X_i, X_{i-1}, and X_{i-2} to determine a corrected peak. If P(J) is a new peak and it is determined that it is a maximum (or minimum), its value must be less (or greater) than P(J-2) to be placed in the P array.

(4) Otherwise P(J) replaces P(J-2) and P(J-2) is stored in an envelope array R(K). A type (b) sequence for the P array is assumed if J reaches at least 5. A type (c) sequence is assumed if K reaches 11 and consecutive R values are monotonic with successive absolute differences becoming smaller.

(5) No matter what type of sequence is finally assumed, the δ^2 transformation is applied to the last five values of the X, P, or R sequence. The resulting three extrapolation values must all be within E of each other for the result to be accepted. Otherwise more X_i are computed.

(6) If the accepted extrapolated values oscillate, the δ^2 transformation is applied to them to get a final answer. Otherwise, the last extrapolated value is taken as the answer.

Although the above procedure is not foolproof, it has rarely failed to give a satisfactory answer, because the magnitude of differences of the final array used in the δ^2 transformation gets smaller, i.e., is nearly geometric.

4. COMPUTING MANY F(t) WITH SUBROUTINE POINTS

Subroutine POINTS computes F(t) at all t supplied in the array T1 (see listing). The T1(I) should be ordered so that the values increase with I (for $I \le 501$).

To save computer time, provision is made to compute some of the answers Y1(I) by interpolation. The user supplies the number N1 of total points (the dimension of T1) and the minimum number N2($\ge$2) that he desires to be computed by integrating the Bromwich integral as described in section 2. POINTS distributes these as equally spaced as possible. From these computed Y1(I), it then attempts interpolation for all other points using subroutine INTERP, based on the routine ALI in the 360 scientific subroutine package.[9] The modifications producing INTERP allow for the natural ordering of the T1 array and do not destroy that array. The listing is fully documented; changes of ALI are identified by the lack of identification in columns 73 to 80.

If INTERP cannot obtain an accurate enough interpolated value (using the absolute error requirement specified by the user) for any reason, the point is calculated by calling TPOINT. Thus, a dense array T1 can be specified (say for plotting purposes), but only as many points as will be required will be calculated by integration.

Provision is also made to enter data (as is required when $t \leq 0$) and to force integration of the Bromwich integral for any point, by means of the I/O array IE. See the listing for a description of its use. Thus, suspect points computed by interpolation can be recomputed.

Another provision is made for printing the results either as they are computed (*not* in the order of increasing T1) or in ordered form after all computations are finished. Printing can also be suppressed. The variable IPRINT controls these features.

5. EXAMPLES

Three examples are given to demonstrate the use of POINTS.

5.1 Example 1

We do first a trivial example

$$f(s) = \frac{s^4 - 6s^2 + 1}{(s^2 + 1)^4} , \tag{31}$$

the solution of which is $F(t) = t^3 (\cos t)/6$. We wish the func-ional values for the range $0 \leq t \leq 10$. Since the maximum magnitude of the function is about 200 in this range, a good graph of the function is obtained with an absolute error specification of 0.05. About 200 points provide a dense enough grid for this case (we use 201 to get $\Delta t = 0.05$) and we limit the number of intervals to integrate to 100 per point because F(t) is so smooth. A main program to print points as they are calculated (IPRINT = 1) is shown.

---

[9] *"System/360 Scientific Subroutine Package, Version III, Programmer's Manual," Application Program GH20-0205-4 (1968), pp. 241-242.*

```
      EXTERNAL T1
      DIMENSION W1(100), T(201), Y(201), IE(201)
      DO 5 I = 1, 201
         IE(I) = 0
    5 T(I) = FLOAT (I-1)/20.
      IE(1) = 1
      Y(1)  = 0.
      CALL POINTS (T1,0.,.05,100,W1,201,21,T,Y,IE,1)
      STOP
      END
```

Note that

(a) The IE and T arrays must be defined before POINTS is called; IE(I) = 0 implies that T(I) must be calculated.

(b) Since T(1) = 0, Y(1) must be supplied. This can usually be found easily from the initial value theorem $[F(0) = \lim_{s\to\infty} sf(s)]$. Y(1) is set to zero, and IE(1) = 1 to indicate that the data are supplied.

(c) N2 = 21 was chosen arbitrarily as the number of points to be computed by integration initially. (Of course, if supplied with IE(I) = 1 the integration is bypassed.)

(d) The Laplace transform is called T1 and since its poles are $0 \pm i$, zero is specified as P.

Running times will be proportional to the time of computing f(s), so any procedure to speed the process is helpful. The Laplace transform routine T1 was written with this in mind:

```
      SUBROUTINE T1(S,F)

      COMPLEX S, F, S2, S1

      S2 = S*S

      S1 = S2 + 1.

      S1 = S1*S1

      S1 = S1*S1

      F  = (1. + S2*(S2 - 6.))/S1

      RETURN

      END
```

Figure 3 (p.19) shows a partial output. Point numbers 1 through 201 in steps of 10 (uncaptioned column at left) are computed first. Point 1 was supplied and not computed by integration so that IE(1) was returned equal to 1. The other points were computed by integration since $IE(1) \geq 11$. The fact that IE(1) was 11 or 12 shows that convergence was very fast.

POINTS then tries to interpolate other points and succeeds except for points 146 and 156, which are computed next by integration. The program then reinterpolates (using the new data also) and starts listing the remaining points with IE = 5. The entire process took about 2 s on an IBM 360/195 computer.

5.2 Example 2

We invert equation (2) with $G(s) = \sqrt{s^2 + s}$. The main program is similar to example 1, except that the call to POINTS will be

```
      CALL POINTS(TRANS,0.,.001,300,W1,201,41,T,Y,IE,2).
```

Because of knowledge of the system, discontinuities are expected and 41 initial points are requested by integration, each point being allowed to sum and extrapolate 300 intervals. The error parameter was chosen to be 0.001 to obtain graphical accuracy, since it is known that the steady state value is one. IPRINT was set to 2 to list all values in sequence.

```
          TIME              FUNCTION   ERROR CODE


   1   0.0                0.0                 1
  11   5.0000000E-01   1.8282734E-02      11
  21   1.0000000E 00   9.0070307E-02      11
  31   1.5000000E 00   3.9605659E-02      11
  41   2.0000000E 00  -5.5468718E-01      11
  51   2.5000000E 00  -2.0863752E 00      11
  61   3.0000000E 00  -4.4551315E 00      11
  71   3.5000000E 00  -6.6920481E 00      11
  81   4.0000000E 00  -6.9725943E 00      11
  91   4.5000000E 00  -3.2058296E 00      11
 101   5.0000000E 00   5.9134254E 00      12
 111   5.5000000E 00   1.9674393E 01      12
 121   6.0000000E 00   3.4569290E 01      11
 131   6.5000000E 00   4.4702072E 01      12
 141   7.0000000E 00   4.3104736E 01      12
 151   7.5000000E 00   2.4376037E 01      12
 161   8.0000000E 00  -1.2417366E 01      12
 171   8.5000000E 00  -6.1632935E 01      12
 181   9.0000000E 00  -1.1072433E 02      12
 191   9.5000000E 00  -1.4251685E 02      12
 201   1.0000000E 01  -1.3987590E 02      12
 146   7.2500000E 00   3.6076628E 01      12
 156   7.7500000E 00   8.0533657E 00      12
   2   4.9999997E-02  -5.7944469E-04       5
   3   9.9999964E-02  -6.2283944E-04       5
   4   1.4999998E-01  -1.3319403E-04       5
   5   1.9999999E-01   8.9251250E-04       5
   6   2.5000000E-01   2.4532527E-03       5
   7   2.9999995E-01   4.5490488E-03       5
   8   3.4999996E-01   7.1798936E-03       5
   9   3.9999998E-01   1.0345794E-02       5
  10   4.4999999E-01   1.4046736E-02       5
  12   5.4999995E-01   2.3053758E-02       5
  13   5.9999996E-01   2.8359649E-02       5
  14   6.4999998E-01   3.4200989E-02       5
  15   6.9999999E-01   4.0577170E-02       5
  16   7.5000000E-01   4.7488410E-02       5
  17   7.9999995E-01   7.6001465E-02       5
```

Figure 3. Partial output of example 1.

The transform subroutine is trivial to write:

```
SUBROUTINE TRANS(S,F)

COMPLEX S,F,E

E = CEXP(CSQRT(S*S + S))

F = 2./(S*(E + 1./E))

RETURN

END
```

Figure 4 shows a partial output with the point numbers listed in the proper sequence. All points that were computed by integration were done successfully (in less than 300 intervals) with CPU time totalling about 17 s on an IBM 360/195. The maximum number of intervals required was 284 for point 103 ($t = 5.1$). Figure 5 shows a graph of the computed function. There actually are discontinuities at every odd integer value of time, and the adaptive procedure clusters many calculations about 1, 3, and 5. The interpolated values at $t = 0.95$ and $t = 4.95$ were accepted by POINTS and are in error. Similarly, points near $t = 7$ and $t = 9$ were computed by interpolation and are in error. Any suspect points can be forced to be computed by integration with the input IE = 3 (see pp. 21 and 22 for fig. 4 and 5).

5.3 Example 3

As a final example, we invert a much more complex but physically realistic case for equation (2), with

$$G(s) = s\left[\left(\frac{1}{1 - \dfrac{2J_1(F)}{FJ_0(F)}}\right)\left(1 + \frac{.8J_1(\sqrt{.71}F)}{\sqrt{.71}FJ_0(\sqrt{.71}F)}\right)\right]^{\frac{1}{2}}, \tag{32}$$

where

$$F = i\sqrt{8s}, \qquad i = \sqrt{-1}, \tag{33}$$

and $J_0(A)$ and $J_1(A)$ are Bessel functions of the complex argument A. G(s) can be shown to be real when s is real; it is a legitimate transform with a real inverse.

```
         TIME            FUNCTION   ERROR CODE

 1  0.0              0.0                1
 2  4.9999997E-02  2.3233570E-04      5
 3  9.9999964E-02  4.9645780E-04      5
 4  1.4999998E-01  7.9236645E-04      5
 5  1.9999999E-01  1.1200621E-03      5
 6  2.5000000E-01  1.4795458E-03     11
 7  2.9999995E-01  1.8708138E-03      5
 8  3.4999996E-01  2.2938703E-03      5
 9  3.9999998E-01  3.7267059E-03      5
10  4.4999999E-01  3.8873379E-03      5
11  5.0000000E-01  3.7537606E-03     11
12  5.4999995E-01  5.9072219E-05     17
13  5.9999996E-01  7.7598752E-04     11
14  6.4999998E-01  1.0814792E-03     12
15  6.9999999E-01  2.7733436E-04     17
16  7.5000000E-01  1.9084013E-03     15
17  7.9999995E-01  3.3322402E-05     15
18  8.4999996E-01  6.8813126E-05     21
19  8.9999998E-01  1.5895028E-05     32
20  9.4999999E-01  2.0178050E-01      5
21  1.0000000E 00  6.0782605E-01     11
22  1.0499992E 00  1.2206039E 00     75
23  1.0999994E 00  1.2278671E 00     40
24  1.1499996E 00  1.2347345E 00      5
25  1.1999998E 00  1.2412090E 00      5
26  1.2500000E 00  1.2472906E 00     19
27  1.2999992E 00  1.2535877E 00      5
28  1.3499994E 00  1.2597990E 00      5
29  1.3999996E 00  1.2659225E 00      5
30  1.4499998E 00  1.2716322E 00      5
31  1.5000000E 00  1.2779064E 00     18
32  1.5499992E 00  1.2842531E 00      5
33  1.5999994E 00  1.2906771E 00      5
34  1.6499996E 00  1.2981052E 00      5
35  1.6999998E 00  1.3043680E 00      5
36  1.7500000E 00  1.3103981E 00     22
37  1.7999992E 00  1.3161907E 00      5
38  1.8499994E 00  1.3217497E 00      5
39  1.8999996E 00  1.3265905E 00      5
40  1.9499998E 00  1.3318396E 00      5
41  2.0000000E 00  1.3370161E 00     11
42  2.0499992E 00  1.3421164E 00      5
```

Figure 4. Partial output of example 2.

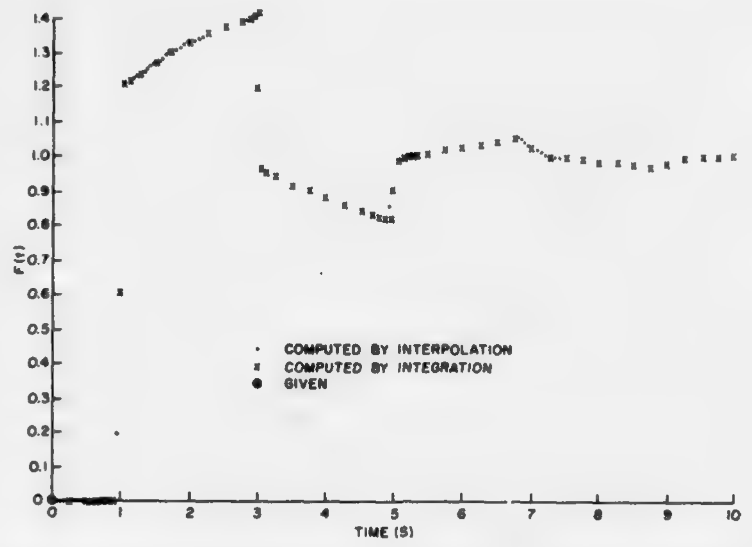


Figure 5. Graph of inverse of $f(s) = 1/[s \cosh(\sqrt{s^2 + s})]$.

Because of the time required to compute the needed Bessel functions, the call to POINTS will specify 101 points, 10 of which are initially found by integration and one point, F(0) = 0, is supplied. The main program is otherwise identical to example 2. IPRINT = 1 was used to print the points as they were computed.

The transform routine is written to reflect equations (2), (32), and (33):

```
SUBROUTINE TRANS(S,Z)
COMPLEX S,Z,F,G,E,J0,J1
DATA S71/.84261498/
F = (0.,1.)*CSQRT(8.*S)
CALL CBES01(F,J0,J1)
E = 1./(1. - 2.*J1/(F*J0))
F = F*S71
CALL CBES01(F,J0,J1)
G = 1. + .8*J1/(F*J0)
G = S*CSQRT(E*G)
E = CEXP(G)
Z = 2./(S*(E + 1./E))
RETURN
END
```

where subroutine CBESO1(F,J0,J1) was a local program to compute the Bessel functions $J_0(F)$ and $J_1(F)$ by a series expansion.

Figure 6 shows a partial output of the computed points, taking just 24 s of CPU time on an IBM 360/195. All the points not shown were computed by interpolation.

```
           TIME              FUNCTION   ERROR CODE

  1   0.0               0.0                 1
 11   1.0000000E 00  -6.1158224E-03        11
 21   2.0000000E 00   1.1911774E 00        12
 31   3.0000000E 00   1.3226652E 00        16
 41   4.0000000E 00   1.0734015E 00        12
 51   5.0000000E 00   9.4529492E-01        14
 61   6.0000000E 00   9.3680280E-01        17
 71   7.0000000E 00   9.8935735E-01        15
 81   8.0000000E 00   1.0182199E 00        18
 91   9.0000000E 00   1.0121737E 00        21
101   1.0000000E 01   1.0001564E 00        23
  6   5.0000000E-01   5.9992250E-05        15
 16   1.5000000E 00   1.0198050E 00        11
 26   2.5000000E 00   1.2702408E 00        11
 36   3.5000000E 00   1.2158699E 00        14
 56   5.5000000E 00   9.1719800E-01        16
 66   6.5000000E 00   9.6558511E-01        18
 18   1.6999998E 00   1.1101294E 00        12
 23   2.1999998E 00   1.2279797E 00        11
 28   2.6999998E 00   1.2932730E 00        14
 33   3.1999998E 00   1.3228388E 00        16
 38   3.6999998E 00   1.1484756E 00        15
 43   4.1999998E 00   1.0367432E 00        12
 53   5.1999998E 00   9.2949528E-01        15
  9   7.9999995E-01   3.9430073E-05        11
 12   1.0999994E 00   4.2133701E-01        32
 14   1.2999992E 00   8.5596758E-01        14
 17   1.5999994E 00   1.0704069E 00        11
 10   8.9999998E-01   3.5557678E-05        22
  2   9.9999964E-02   2.1424683E-05         5
  3   1.9999999E-01   3.8136262E-05         5
  4   2.9999995E-01   5.0134695E-05         5
  5   3.9999998E-01   5.7419995E-05         5
  7   5.9999996E-01   5.1647352E-05         5
  8   6.9999999E-01   4.4793269E-05         5
 13   1.1999998E 00   7.0375639E-01        19
 15   1.3999996E 00   9.5463085E-01         5
```

Figure 6. Partial output of example 3 (cont'd).

```
19   1.7999992E 00   1.1420450E 00    5
20   1.8999996E 00   1.1685181E 00    5
22   2.0999994E 00   1.2108688E 00    5
24   2.2999992E 00   1.2434921E 00    5
25   2.3999996E 00   1.2574787E 00    5
27   2.5999994E 00   1.2822704E 00    5
29   2.7999992E 00   1.3037567E 00    5
30   2.8999996E 00   1.3145561E 00    5
32   3.0999994E 00   1.3266182E 00    5
34   3.2999992E 00   1.3017693E 00    5
35   3.3999996E 00   1.2507420E 00    5
37   3.5999994E 00   1.1817799E 00    5
39   3.7999992E 00   1.1203184E 00    5
40   3.8999996E 00   1.0954034E 00    5
```

Figure 6. Partial output of example 3.

Figure 7 shows a graph of the results. The smoothness of the computed points suggests that good graphical accuracy has been obtained.

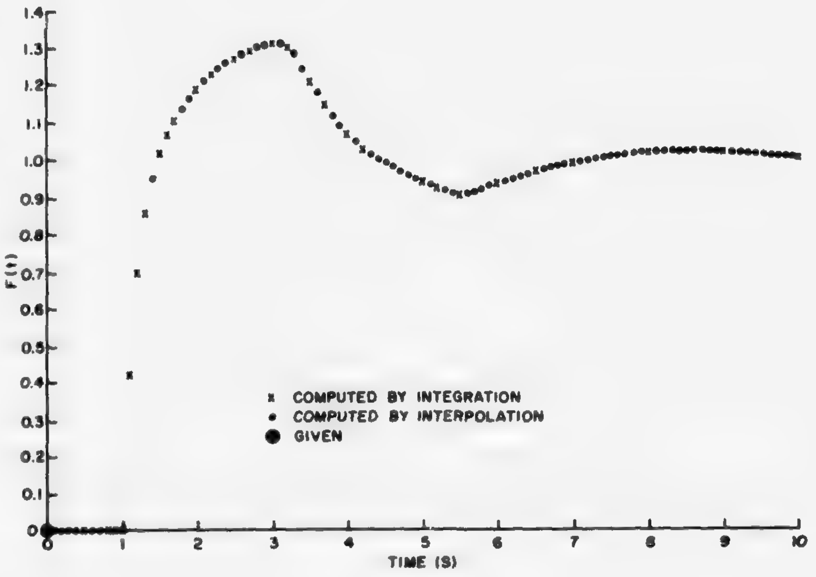


Figure 7. Graph of example 3.

6. CONCLUSIONS

On the basis of the three examples given here and others tried, subroutine POINTS, in conjunction with others listed in the appendix, is an effective method for inverting a Laplace transform. The adaptive interpolative procedure helps cut computer time by not requiring all points to be computed by directly computing the Bromwich integral. Thus, costs are nominal even for relatively complicated transforms. Since the use of the program is relatively easy, it is deemed a useful asset to any software library.

LITERATURE CITED

(1) C. Lanczos, Applied Analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ (1956).

(2) A. Papoulis, "A New Method of Inversion of the Laplace Transform," Quarterly of Applied Mathematics, 14 (1957).

(3) B. S. Berger, "Inversion of the N-Dimensional Laplace Transform," Mathematics of Computation, 20 (1966).

(4) R. Bellman, R. E. Kalaba, and J. A. Lockett, Numerical Inversion of the Laplace Transform, American Elsevier Publishing Co., New York (1966).

(5) B. S. Berger and S. Duangudom, "A Technique for Increasing the Accuracy of the Numerical Inversion of the Laplace Transform with Applications," ASME Journal of Applied Mechanics, paper No. 73-WA/APM-1 (December 1973).

(6) L. A. Schmittroth, "Numerical Inversion of Laplace Transforms," ACM Communications (March 1960).

(7) F. Scheid, Theory and Problems of Numerical Analysis, McGraw-Hill Book Co., Shaum's Outline Series, New York (1968).

(8) R. V. Churchill, Complex Variables and Applications, 2nd ed., McGraw-Hill Book Co., New York (1960).

(9) "System/360 Scientific Subroutine Package, Version III Programmer's Manual," Application Program GH20-0205-4 (1968).

(10) D. Shanks, "Non-Linear Transformation of Divergent and Slowly Convergent Sequences," Journal of Mathematical Physics, 34 (1955).

APPENDIX A.--PROGRAM LISTING

This appendix contains the listing of the subroutine POINTS and all subroutines it calls with the exception of TRANS. The user must write a main program that calls POINTS (or TPOINT if only one point is to be calculated) and the subroutine TRANS.

Preceding page blank

APPENDIX A

```
C.................................................................
      SUBROUTINE POINTS(TRANS,P,    E,M,W1,    N1,N2,T1,Y1,IE,IPRINT)
C  THIS ROUTINE COMPUTES THE INVERSE LAPLACE TRANSFORM FOR A SET OF
C  POINTS SPECIFIED BY THE USER.  THE ARGUMENT LIST IS DEFINED AS
C  FOLLOWS.
C
C      TRANS - THE DUMMY NAME OF THE SUBROUTINE THAT COMPUTES THE
C              TRANSFER FUNCTION VALUES, AND WHICH MUST BE WRITTEN BY
C              THE USER. ITS EXACT FORM IS
C                   SUBROUTINE TRANS(S,F)
C                   COMPLEX S,F
C              FOR ANY COMPLEX INPUT S, F = F(S) MUST BE CALCULATED
C              WITHOUT DESTROYING S. THE ACTUAL NAME OF THIS SUBROUTINE
C              MUST BE DECLARED TO BE EXTERNAL BY THE CALLING PROGRAM.
C          P - THE MAXIMUM OF THE REAL PARTS OF ALL SINGULARITIES OF
C              TRANS. THESE INCLUDE POLES, BRANCH POINTS, AND ESSENTIAL
C              SINGULARITIES.
C          E - THE ABSOLUTE ERROR DESIRED FOR THE POINTS CALCULATED.
C              THIS IS MORE OF A GUIDE, SINCE THE ACTUAL ERROR MAY BE
C              SLIGHTLY GREATER.
C          M - THE MAXIMUM NUMBER OF 2.*PI INTERVALS CONSIDERED. IF F(T)
C              IS CONTINUOUS WITH CONTINUOUS DERIVATIVES, 100 IS USUALLY
C              SUFFICIENT.
C         W1 - A WORKSPACE DIMENSIONED M IN THE CALLING PROGRAM.
C         N1 - THE DIMENSION OF T1, Y1, AND IE BELOW. THIS IS THE NUMBER
C              OF POINTS TO BE COMPUTED. SHOULD BE .GE.2 AND .LE.501
C         N2 - THE NUMBER OF POINTS INITIALLY COMPUTED BY INTEGRATION
C              OF THE BROMWICH INTEGRAL (N2.GE.2). IF THE FUNCTION IS
C              SMOOTH, N2=5 OR 10 IS USUALLY SUFFICIENT.THE POINTS
C              COMPUTED ARE APPROXIMATELY EQUISPACED OVER N1.
C         T1 - THE ARRAY OF TIMES FOR WHICH THE INVERSE TRANSFORM IS
C              DESIRED. THE VALUES OF T1 MUST BE STORED PRIOR TO CALLING
C              POINTS AND ORDERED TO INCREASE WITH I.
C         Y1 - THE ARRAY OF ANSWERS. Y1(I) CORRESPONDS TO T1(I).
C         IE - INPUT/OUTPUT ARRAY.  AS INPUT,
C                   IE(I) = 0 MEANS THAT Y1(I) MUST BE COMPUTED BY THE
C                             PROGRAM AS IT SEES FIT.
C                         = 1 MEANS THAT Y1(I) IS SUPPLIED. (IF T1(I)
C                             .LE.0, Y1(I) MUST BE SUPPLIED.)
C                         = 3 IS A REQUEST TO COMPUTE Y1(I) VIA THE
C                             BROMWICH INTEGRAL.
C              AS OUTPUT,
C                   IE(I) = L (.GE.11) MEANS Y1(I) WAS SUCCESSFULLY
C                             COMPUTED VIA THE BROMWICH INTEGRAL WITH L
C                             INTERVALS OF LENGTH 2.*PI.
C                         = 5 MEANS Y1(I) WAS COMPUTED BY INTERPOLATION.
C                         = 2 MEANS Y1(I) WAS NOT FOUND TO THE ACCURACY
C                             DESIRED IN M INTERVALS OF LENGTH 2.*PI. THE
C                             BEST POSSIBLE ANSWER IS RETURNED.
C                         = 1 MEANS THAT Y1(I) WAS SUPPLIED.
```

```
C                              = -L MEANS THAT THE COMPUTATICN OF POINT I WAS
C                                   ABORTED BECAUSE THE LTH INTERVAL CCULD NOT
C                                   BE INTEGRATED
C     IPRINT - THE PRINT CONTROL.
C              IF IPRINT = 0, NO OUTPUT IS PRINTED.
C                        = 1, PRINTED AS POINTS ARE CCMPUTED (NOT
C                                   ORDERED).
C                        = 2, PRINTING AFTER ALL PCINTS ARE COMPUTED
C                                   (ORDERED).
C
C  ANY POINT COMPUTED BY INTEGRATING THE BROMWICH INTEGRAL IS DONE BY
C  CALLING TPOINT.
C
C  INTERPOLATION IS DONE BY SUBROUTINE INTERP, WHICH IS A MODIFIED
C  ROUTINE FROM THE IBM 360 SSP.
C
      DIMENSION W1(1),      T1(1),Y1(1),IE(1),T(501),Y(501),ARG(501),
     * VAL(501)
      EXTERNALTRANS
C  PRINT HEADER IF REQUIREC.
      IF(IPRINT.GE.1) WRITE(6,22)
C
C  SET UP THE N1 POINTS TO BE CALCULATED BY INTEGRATION. J IS THE INDEX
C  OF THE POINT, FIRST COMPUTED IN FLOATING POINT BY FJ, AND INCREMENTED
C  BY XN.
      FJ=1.
      J=1
      XN=FLOAT(N1-1)/FLOAT(N2-1)
    3 DO 5 I=1,N2
      IF(I.EC.N2) J=N1
      IF(IE(J).EQ.1) GOTO 7
      IF(IE(J).NE.0) GOTO 6
      IF(T1(J).GT.0.) GOTO 4
      IE(J)=-1
      GOTO 7
    4 CALL TPOINT(TRANS,P,T1(J),     Y1(J),E,M,IE(J),W1)
      IF(IE(J).EQ.0) IE(J)=2
    7 IF(IPRINT.EQ.1) WRITE(6,24) J,T1(J),Y1(J),IE(J)
    6 FJ=FJ+XN
      J=FJ+.5
    5 CONTINUE
C  COMPUTE SPECIAL PCINTS REQUESTED (IE(J) = 3).
      DO 8 J=1,N1
      IF(IE(J).NE.3) GOTO 8
      IF(T1(J).LE.0.) GOTO 8
      CALL TPOINT(TRANS,P,T1(J),     Y1(J),E,M,IE(J),W1)
      IF(IE(J).EQ.0) IE(J)=2
      IF(IPRINT.EQ.1) WRITE(6,24) J,T1(J),Y1(J),IE(J)
    8 CONTINUE
C  FIND INDICES I1 AND I2 SURROUNDING UNCOMPUTED POINTS.
```

APPENDIX A

```
    9 I2=1
      NN=1
   10 J=0
      DO 20 I=I2,N1
      IF(J.NE.0) GOTO 15
      IF(IE(I).NE.0.AND.IE(I+1).EQ.0) GOTO 18
      GO TO 20
   15 IF(IE(I).NE.0.AND.IE(I-1).EQ.0) GOTO 17
      GOTO 20
   17 I2=I
      GOTO 25
   18 J=1
      I1=I
   20 CONTINUE
      IF(I2.NE.1) GOTO 9
      GOTO 70
C     CHOOSE K ABOUT MIDWAY BETWEEN I1 AND I2.
   25 K=(I1+I2)/2
C     MOVE ALL GOOD POINTS TO T AND Y AND TRY TO INTERPOLATE POINT K.
      J=0
      DO 30 I=1,N1
      IF((IE(I).NE.1) .AND. (IE(I).LT. 9)) GOTO 30
      J=J+1
      T(J)=T1(I)
      Y(J)=Y1(I)
   30 CONTINUE
C     J IS THE NUMBER OF POINTS. IF LESS THAN 5, CALL TPOINT.
      IF(J.LT.5) GOTO 50
      CALL INTERP(J,T,Y,T1(K),Y1(K),E,IE(K),NN,ARG,VAL)
      IF(IE(K).NE.0) GOTO 50
      IE(K)=5
      GOTO 10
C     POINT COULD NOT BE INTERPOLATED. CALL TPOINT.
   50 CALL TPOINT(TRANS,P,T1(K),     Y1(K),E,M,IE(K),W1)
      IF(IE(K).EQ.0) IE(K)=2
   60 IF(IPRINT.EQ.1) WRITE(6,24) K,T1(K),Y1(K),IE(K)
      GOTO 10
C     LAST PASS. MOVE ALL GOOD POINTS TO T AND Y.
   70 J=0
      DO 80 I=1,N1
      IF(IE(I).NE.1.AND.IE(I).LT.6) GOTO 80
      J=J+1
      T(J)=T1(I)
      Y(J)=Y1(I)
   80 CONTINUE
      NN=1
      DO 90 K=1,N1
      IF(IE(K).EQ.1.OR.IE(K).GT.5) GOTO 90
      IF(J.LT.5) GOTO 95
      CALL INTERP(J,T,Y,T1(K),YTEST,E,IERR,NN,ARG,VAL)
      IF(IERR.NE.0) GOTO 95
      Y1(K)=YTEST
      IE(K)=5
      GOTO 98
   95 IF(IE(K).EQ.2.OR.IE(K).LT.0) GOTO 90
      CALL TPOINT(TRANS,P,T1(K),     Y1(K),E,M,IE(K),W1)
      IF(IE(K).EQ.0) IE(K)=2
   98 IF(IPRINT.EQ.1) WRITE(6,24) K,T1(K),Y1(K),IE(K)
   90 CONTINUE
      IF(IPRINT.LE.1) RETURN
C  PRINT OUTPUT IF IPRINT IS GREATER THAN 1.
   22 FORMAT(1H1,9X,4HTIME,8X,8HFUNCTION,2X,10HERRCR CCDE /)
      DO 23 I=1,N1
   23 WRITE(6,24) I,T1(I),Y1(I),IE(I)
   24 FORMAT(I4,1P2E15.7,I5)
      RETURN
      END
C..........................................................................
      SUBROUTINE TPCINT(TRANS,P,T,     Y,EPS,M,IE,W1)
C  TPOINT COMPUTES CNE PCINT Y(T) OF THE LAPLACE TRANSFCRM INVERSE BY
C  INTEGRATING THE BROMWICH INTEGRAL. THE APGUMENT LIST IS AS FOLLOWS.
C
C      TRANS - SEE POINTS FOR DEFINITION
C          P - SEE POINTS FOR DEFINITION
C          T - THE TIME FOR WHICH THE INVERSE LAPLACE TRANSFORM IS
C              DESIRED
C          Y - THE ANSWER
C        EPS - THE ABSOLUTE ERROR DESIRED. THIS IS MORE OF A GUIDE,
C              SINCE THE ACTUAL ERROR MAY BE SLIGHTL. GREATER.
C          M - SEE PCINTS FCR DEFINITION
C         IE - OUTPUT ERROR CODE.
C                    IE = 0 MEANS THAT THE ACCURACY COULD NCT BE OBTAINED
C                            IN M INTERVALS OF WIDTH 2.*PI.  Y IS THE BEST
C                            ANSWER OBTAINED.
C                       = L (.GT. 0) MEANS THAT THE ANSWER WAS OBTAINED
C                            IN JUST L 2.*PI INTERVALS.
C                       = -L (.LT. 0) MEANS THAT THE RUN WAS ABORTED
C                             BECAUSE OF DIFFICULTY INTEGRATING INTERVAL L.
C         W1 - WORKSPACE OF DIMENSION M.
C
C  INTEGRATION OF THE BRCMWICH INTEGRAL IS DONE BY SUBROUTINE SUM. IT
C  SUMS THE RESULTS CF INTEGRALS FOR INTERVALS OF 2.*PI AND ATTEMPTS TO
C  ACCELERATE THE SUM BY CERTAIN TRANSFORMATIONS. THE BRCMWICH INTEGRAL
C  IS COMPUTED ALONG A PATH S = GC/T + P WHERE GC IS INITIALLY .01.
C  THIS IS CLOSE TO A SINGULARITY AND CONVERGES IN SUM RAPIDLY. IF
C  DIFFICULTY IS ENCOUNTERED IN ANY INTERVAL BECAUSE GC IS TOO SMALL,
C  GC IS DOUBLED, FOLLOWED BY ANOTHER ATTEMPT TO COMPUTE THE INTEGRAL.
      EXTERNAL TRANS
      COMMON/TPT/G,N1,E,T1
      COMMON/Q/PI,PI2,XX,YY
```

APPENDIX A

```
      DIMENSION W1(1)
      IF(M.LT.1) RETURN
      PI=3.141593
      PI2=6.283185
      IE=-1
      T1=T
      GC=.01
C  N1 CONTROLS THE GAUSSIAN FORMULA USED FOR THE FIRST INTERVAL. SET TO
C  5 TO USE A 32-POINT FORMULA.
      N1=5
    2 G=GC/T+P
C  ABORT IF G*T IS TOO LARGE.
      IF(G*T.GT.11.) RETURN
C  EGT IS THE COMMON FACTOR OF ALL TERMS SUMMED.
      EGT=2.*EXP(G*T)/T
C  ADJUST THE ERROR REQUIREMENT FOR EGT AND PASS E IN COMMON.
      E=EPS/EGT
      CALL  SUM(    Y,M,E,IE,TRANS,K,W1)
      IF(IE.LT.0) GOTO 4
      IF(IE.GT.0) GOTO 3
C  ANSWER IS OK, SET IE TO K, THE NUMBER OF INTERVALS NEEDED.
      IE=K
    1 Y=Y*EGT
      RETURN
C  ANSWER NOT FOUND WITHIN M INTERVALS. FLAG WITH IE=0.
    3 IE=0
      GOTO 1
C  SUM COULD NOT INTEGRATE ONE OF THE INTERVALS. DOUBLE CG AND TRY AGAIN
    4 GC=GC+GC
      GOTO 2
      END
C.........................................................................
      SUBROUTINE SUM(    Y,M,E,IE,TRANS,K,X)
C
C  SUM COMPUTES THE BROMWICH INTEGRAL BY OBTAINING A SEQUENCE OF
C  PARTIAL SUMS AND APPLYING THE DELTA SQUARE TRANSFORMATION
C        1. TO THE SEQUENCE, IF IT MONOTONIC WHERE THE MAGNITUDE OF THE
C           DIFFERENCE IS DECREASING 10 CONSECUTIVE TIMES
C        2. TO A SUBSEQUENCE OF PEAKS WHERE THE MAXIMUMS ARE DECREASING
C           AND MINIMUMS ARE INCREASING
C        3. TO A SUBSEQUENCE OF AN ENVELOPE OF THE PEAKS WHERE THE
C           MAXIMUMS ARE INCREASING OR THE MINIMUMS ARE DECREASING
C  IF 3 CONSECUTIVE PROJECTIONS ARE WITHIN E OF EACH OTHER, THE SUM IS
C  CONSIDERED FOUND. IF THE PROJECTIONS OSCILLATE, A DELTA SQUARE
C  TRANSFORMATION IS APPLIED TO THEM TO OBTAIN A FINAL ANSWER. OTHER-
C  WISE THE LAST PROJECTION IS ACCEPTED AS THE FINAL ANSWER.
C  THE ARGUMENT LIST IS AS FOLLOWS
C          Y - THE BROMWICH INTEGRAL ANSWER (WITHOUT THE FACTOR EGT IN
C              TPOINT).
C          M - SEE POINTS FOR DEFINITION
```

```
C          E - SEE POINTS FOR DEFINITION
C         IE - OUTPUT ERROR CODE
C              IE = 0 MEANS THAT Y WAS FOUND
C                 = M MEANS THAT THE ACCURACY COULD NOT BE ACHIEVED. THE
C                     LAST VALUE OF THE PARTIAL SUM, X(M), IS RETURNED
C                     AS THE FINAL ANSWER.
C                 = -L (.LT. 0) MEANS THAT THE RUN WAS ABORTED BECAUSE
C                      OF DIFFICULTIES IN INTEGRATING INTERVAL L.
C      TRANS - SEE POINTS FOR DEFINITION
C          K - THE NUMBER OF INTERVALS USED IF IE = 0.
C          X - THE WORKSPACE OF DIMENSION M, USED TO STORE THE PARTIAL
C              SUMS OF THE BROMWICH INTEGRAL.
C
      DIMENSION X(1),DE(3),P(11),R(11)
      EXTERNAL TRANS
C  X IS THE ARRAY OF PARTIAL SUMS
C  NR IS THE NUMBER OF ELEMENTS IN THE R ARRAY THAT STORES ENVELOPE
C    INFORMATION OF THE PEAKS OF X.
C  NP IS THE NUMBER OF ELEMENTS IN THE P ARRAY THAT STORES THE PEAKS
C    OF X.
C  ND IS THE NUMBER OF CONSECUTIVE VALUES OF X THAT ARE MONOTONIC WITH
C    DECREASING DIFFERENCES
      NR=0
      NP=0
      ND=0
C  COMPUTE THE INTEGRALS OF EACH INTERVAL IN A DO LOOP.
      DO 105 I=1,M
      K=I
C  D IS THE VALUE OF THE INTEGRAL OF THE CURRENT INTERVAL.
C  D1 IS THE VALUE OF THE PREVIOUS INTERVAL.
C  DA AND DL ARE THE ABSOLUTE VALUES OF D AND D1.
      D=FCT(I,TRANS,IE)
      D2=D
C  ABORT IF THERE IS TROUBLE IN COMPUTING D.
      IF(IE.LT.0) RETURN
      DA=ABS(D)
      IF(I.NE.1) GOTO 5
      X(1)=D
      GOTO 100
C  SET UP X VALUES AND COMPARE DIFFERENCES.
    5 X(I)=X(I-1) + D
      IF(DA.GT.DL) GOTO 10
C  THE DIFFERENCE IS LESS THAN THE PREVIOUS DIFFERENCE.
      ND=ND+1
      IF(I.LT.11) GOTO 100
C  CHECK FOR OSCILLATING SEQUENCE.
      IF( D*D1.LT.0.) GOTO 10
      IF(ND.LT.10) GOTO 20
C  THERE ARE 10 CONSECUTIVE MONOTONIC VALUES OF X(I). EXTRAPOLATE THE
C  LAST 5 VALUES WITH THE DELTA SQUARE TRANSFORMATION.
```

```
      CALL DELTA2(5,X(I-4),DE)
C  REJECT IF ALL 3 EXTRAPOLATED VALUES ARE NOT WITHIN E OF EACH OTHER.
      IF(ABS(DE(1)-DE(2)).GT.E) GOTO 100
      IF(ABS(DE(3)-DE(2)).GT.E) GOTO 100
      IF(ABS(DE(3)-DE(1)).GT.E) GOTO 100
   11 IF((DE(3)-DE(2))*(DE(2)-DE(1)).GT.0.) GOTO 8
C  EXTRAPOLATE THE EXTRAPOLATED VALUES IF THEY OSCILLATE.
      CALL DELTA2(3,DE,DE)
      Y=DE(1)
      GO TO 9
C  ANSWER ACCEPTED.
    8 Y=DE(3)
    9 IE=0
      RETURN
C  THE MAGNITUDE OF THE DIFFERENCES OF THE X(I) ARE INCREASING. CHECK
C  FOR PEAKS
   10 ND=0
      IF(I.LT.11) GOTO 100
   20 II=2
      IF(NP.NE.0) II=IP+1
      IM1=I-1
      DO 15 J=II,IM1
C  D AND D1 CHANGE MEANINGS AND ARE NOW TEMPORARY VARIABLES.
      D=X(J)-X(J-1)
      D1=X(J+1)-X(J)
      IF(D.GE.0.) GOTO 17
      IF(D1.LT.0.) GOTO 15
C  A MINIMUM IS DETECTED. REFINE WITH SUBROUTINE PEAK.
      NP=NP+1
      IP=J
      CALL PEAK(P(NP), X(J),D,D1)
   14 IF(NP.LE.2) GOTO 15
C  REJECT IF MINIMUM IS LESS THAN THE PREVIOUS MINIMUM.
      IF(P(NP).LE.P(NP-2)) GOTO 16
C  REJECT IF X(J+1) IS .GE. THE PREVIOUS MINIMUM.
      IF(X(J+1).LT.P(NP-1)) GO TO 18
      GOTO 23
C  THE PEAK IS REJECTED. STORE IN THE R ARRAY.
   16 NP=NP-2
      NR=NR+1
      R(NR)=(P(NP)+P(NP+1))*.5
      IJ=1
      P(NP)=P(NP+2)
      IF(NR.EQ.11) GOTO 25
C  CONTINUE CHECKING THE NEW PEAK AGAINST THE PREVIOUS ONES.
      GOTO 14
   18 IF(NP.LT.5) GOTO 15
      IF(J.NE.IM1) GOTO 23
C  THERE ARE AT LEAST 5 PEAKS. COMPUTE AND CHECK THE DELTA SQUARE
C  TRANSFORMATION.
```

```
      CALL DELTA2(5,P(NP-4),DE)
C  CHECK THE DIFFERENCE OF TRANSFORMED VALUES.
      IF(ABS(DE(1)-DE(2)).LE.E) GOTO 21
C  ERRORS TOO GREAT. MOVE PEAKS IN P ARRAY IF NP IS .GE. 11.
   23 IF(NP.LE.10) GOTO 15
      DO 22 L=1,10
   22 P(L)=P(L+1)
      NP=10
      GOTO 15
C  CONTINUE CHECKING THE DIFFERENCES OF THE TRANSFORMED VALUES.
   21 IF(ABS(DE(2)-DE(3)).GT.E) GOTO 23
      IF(ABS(DE(1)-DE(3)).GT.E) GOTO 23
C  THE ERROR CRITEREA ARE SATISFIED SO THE ANSWER IS FOUND.
      GOTO 11
   17 IF(D1.GT.0.) GOTO 15
C  A MAXIMUM IS DETECTED. REFINE WITH SUBROUTINE PEAK.
      NP=NP+1
      IP=J
      CALL PEAK(P(NP),X(J),D,D1)
   19 IF(NP.LE.2) GOTO 15
C  REJECT IF THE MAXIMUM IS GREATER THAN THE PREVIOUS ONE.
      IF(P(NP).GE.P(NP-2)) GOTO 24
C  REJECT IF X(J+1) IS .LE. THE PREVIOUS MINIMUM.
      IF(X(J+1).GT.P(NP-1)) GOTO 18
      GOTO 23
C  THE PEAK IS REJECTED. STORE IN THE R ARRAY.
   24 NP=NP-2
      NR=NR+1
      R(NR)=(P(NP)+P(NP+1))*.5
      IJ=2
      P(NP)=P(NP+2)
      IF(NR.EQ.11) GOTO 25
C  CONTINUE CHECKING THE NEW MAXIMUM AGAINST THE PREVIOUS ONES.
      GOTO 19
C  CHECK R ARRAY. IF THE MAGNITUDE OF THE DIFFERENCES IS MONOTONIC
C  DECREASING, THEN EXTRAPOLATE.
   25 DO 26 L=1,9
      IF(ABS(R(L+2)-R(L+1)).GT.ABS(R(L+1)-R(L))) GOTO 27
   26 CONTINUE
C  THE R ARRAY VALUES SEEM TO BE APPROACHING A LIMIT.
      CALL DELTA2(5,R(7),DE)
C  CHECK THE DIFFERENCES OF THE EXTRAPOLATED VALUES FOR ERROR.
      IF(ABS(DE(1)-DE(2)).GE.E) GOTO 27
      IF(ABS(DE(3)-DE(2)).GE.E) GOTO 27
      IF(ABS(DE(3)-DE(1)).GE.E) GOTO 27
C  THE ERROR CRITEREA ARE SATISFIED SO THE ANSWER IS ACCEPTED.
      GOTO 11
C  R ARRAY EXTRAPOLATION YIELDS TOO MUCH ERROR. MAKE ROOM FOR NEXT
C  VALUE.
   27 DO 28 L=1,10
   28 R(L)=R(L+1)
      NR=10
C  CONTINUE SEARCHING FOR PEAKS IN LOOP.
   15 CONTINUE
C  UPDATE AND CONTINUE MAJOR LOOP.
  100 DL=DA
  105 D1=D2
C  THE LIMIT OF THE NUMBER OF INTERVALS IS REACHED. RETURN WITH MTH
C  PARTIAL SUM.
      Y=X(M)
      IE=M
      RETURN
      END
C..........................................................................
C        SUBROUTINE INTERP
C        PURPOSE                                                         ALI  006
C           TO INTERPOLATE FUNCTION VALUE Y FOR A GIVEN ARGUMENT VALUE   ALI  007
C           X USING A GIVEN TABLE (XARG,YVAL) OF ARGUMENT AND FUNCTION
C           VALUES.                                                      ALI  009
C                                                                        ALI  010
C        USAGE                                                           ALI  011
C           CALL INTERP(NDIM,XARG,YVAL,X,Y,EPS,IER,NN,ARG,VAL)
C                                                                        ALI  013
C        DESCRIPTION OF PARAMETERS                                       ALI  014
C           NDIM    - AN INPUT VALUE WHICH SPECIFIES THE NUMBER OF       ALI  021
C                     POINTS IN TABLE (XARG,YVAL).
C           XARG    - THE INPUT VECTOR (DIMENSION NDIM) OF ARGUMENT
C                     VALUES OF THE TABLE (NOT DESTROYED).               ALI  017
C           YVAL    - THE INPUT VECTOR (DIMENSION NDIM) OF FUNCTION
C                     VALUES OF THE TABLE (NOT DESTROYED).
C           X       - THE ARGUMENT VALUE SPECIFIED BY INPUT.             ALI  015
C           Y       - THE RESULTING INTERPOLATED FUNCTION VALUE.         ALI  020
C           EPS     - AN INPUT CONSTANT WHICH IS USED AS UPPER BOUND     ALI  023
C                     FOR THE ABSOLUTE ERROR.                            ALI  024
C           IER     - A RESULTING ERROR PARAMETER.                       ALI  025
C           NN      - AN INPUT ESTIMATE FOR THE SUBSCRIPT TO MAKE
C                     ABS(XARG(NN) - X) A MINIMUM. NN RETURNS SET TO THE
C                     CORRECT VALUE.
C           ARG     - WORKSPACE OF DIMENSION NDIM.
C           VAL     - WORKSPACE OF DIMENSION NDIM.
C                                                                        ALI  026
C        REMARKS                                                         ALI  027
C           (1) TABLE (XARG,YVAL) SHOULD REPRESENT A SINGLE-VALUED
C               FUNCTION AND SHOULD BE STORED SO THAT XARG(I).GE.XARG(J)
C               WHENEVER I.GT.J.
C           (2) NO ACTION BESIDES ERROR MESSAGE IN CASE NDIM LESS        ALI  034
C               THAN 1.                                                  ALI  035
C           (3) INTERPOLATION IS TERMINATED EITHER IF THE DIFFERENCE     ALI  036
C               BETWEEN TWO SUCCESSIVE INTERPOLATED VALUES IS            ALI  037
C               ABSOLUTELY LESS THAN TOLERANCE EPS, OR IF THE ABSOLUTE   ALI  038
```

```
C                  VALUE OF THIS DIFFERENCE STOPS DIMINISHING, OR AFTER
C                  (NDIM-1) STEPS. FURTHER IT IS TERMINATED IF THE
C                  PROCEDURE DISCOVERS TWO ARGUMENT VALUES IN VECTOR ARG
C                  WHICH ARE IDENTICAL. DEPENDENT ON THESE FOUR CASES,
C                  ERROR PARAMETER IER IS CODED IN THE FOLLOWING FORM
C                   IER=0 - IT WAS POSSIBLE TO REACH THE REQUIRED
C                           ACCURACY (NO ERROR).
C                   IER=1 - IT WAS IMPOSSIBLE TO REACH THE REQUIRED
C                           ACCURACY BECAUSE OF ROUNDING ERRORS. INCREASE
C                           EPS AND/OR THE ACCURACY OF THE TABLE.
C                   IER=2 - IT WAS IMPOSSIBLE TO CHECK ACCURACY BECAUSE
C                           NDIM IS LESS THAN 3, OR THE REQUIRED ACCURACY
C                           COULD NOT BE REACHED BY MEANS OF THE GIVEN
C                           TABLE. NDIM SHOULD BE INCREASED.
C                   IER=3 - THE PROCEDURE DISCOVERED TWO ARGUMENT VALUES
C                           IN VECTOR ARG WHICH ARE IDENTICAL.
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           NONE
C
C        METHOD
C           INTERPOLATION IS DONE BY MEANS OF AITKENS SCHEME OF
C           LAGRANGE INTERPOLATION. ON RETURN Y CONTAINS AN INTERPOLATED
C           FUNCTION VALUE AT POINT X, WHICH IS IN THE SENSE OF REMARK
C           (3) OPTIMAL WITH RESPECT TO GIVEN TABLE. FOR REFERENCE, SEE
C           F.B.HILDEBRAND, INTRODUCTION TO NUMERICAL ANALYSIS,
C           MCGRAW-HILL, NEW YORK/TORONTO/LONDON, 1956, PP.49-50.
C
      SUBROUTINE INTERP(NDIM,XARG,YVAL,X,Y,EPS,IER,NN,ARG,VAL)
C
      DIMENSION XARG(1),YVAL(1),ARG(1),VAL(1)
C
      IER=2
      IF(NDIM-1)9,37,31
C
C     FIND THE CORRECT VALUE OF NN
   31 II=NN
      DELT2=X-XARG(II)
C
C     START SEARCH LOOP
      DO 25 K=1,NDIM
      JJ=II
      IF(DELT2)22,23,24
C
C     IF DELT2.EQ.0 THEN NN = II AND Y = YVAL(NN) EXACTLY
   23 NN=II
      Y=YVAL(NN)
      IER=0
      RETURN
C
```

```
C     IF DELT2.GT.O THEN TRY LARGER SUBSCRIPT
   24 II=II+1
      IF(II.LE.NDIM) GO TO 26
      JJ=NDIM
      GO TO 27
C
C     IF DELT2.LT.O THEN TRY SMALLER SUBSCRIPT
   22 II=II-1
      IF(II.GE.1) GO TO 26
      JJ=1
      GO TO 27
   26 DELT1=DELT2
      DELT2=X-XARG(II)
C
C     COMPARE DELT2 WITH DELT1. IF GREATER, THEN CLOSEST POINT IS PAST.
      IF(ABS(DELT2).GE.ABS(DELT1)) GO TO 27
   25 CONTINUE
   27 NN=JJ
      II=JJ
      ARG(1)=XARG(II)
      VAL(1)=YVAL(II)
      DELT2=0.                                                    ALI  073
      J=1
C
C     TRANSFER TO DETERMINE THE SECOND CLOSEST POINT.
      GO TO 6
C                                                                 ALI  075
C     START OF AITKEN-LOOP                                        ALI  076
    1 DELT1=DELT2
      IEND=J-1                                                    ALI  079
      DO 2 I=1,IEND                                               ALI  080
      H=ARG(I)-ARG(J)                                             ALI  081
      IF(H)2,13,2                                                 ALI  082
    2 VAL(J)=(VAL(I)*(X-ARG(J))-VAL(J)*(X-ARG(I)))/H              ALI  083
      DELT2=ABS(VAL(J)-VAL(IEND))                                 ALI  084
      IF(J-3)6,3,3
    3 IF(DELT2-EPS)10,10,4                                        ALI  086
    4 IF(J-5)6,5,5                                                ALI  087
    5 IF(DELT2-DELT1)6,11,11                                      ALI  088
    6 J=J+1
C                                                                 ALI  091
C     END OF AITKEN-LOOP BUT WE MUST FIND THE JTH CLOSEST POINT BEFORE
C     LOOPING BACK TO STATEMENT 1.
      IF(J.GT.NDIM) GO TO 36
      IF(II.EQ.1) GO TO 30
      IF(JJ.EQ.NDIM) GO TO 29
      IF(ABS(XARG(II-1)-X).GT.ABS(XARG(JJ+1)-X)) GO TO 30
   29 II=II-1
      ARG(J)=XARG(II)
      VAL(J)=YVAL(II)
      GO TO 1
   30 JJ=JJ+1
      ARG(J)=XARG(JJ)
      VAL(J)=YVAL(JJ)
      GO TO 1
C
C     DEFINE VAL(1) IN CASE NDIM = 1.
   37 VAL(1) = YVAL(1)
   36 J=NDIM
    8 Y=VAL(J)
    9 RETURN
C
C     THERE IS SUFFICIENT ACCURACY WITHIN NDIM-1 ITERATION STEPS
   10 IER=0
      GOTO 8
C
C     TEST VALUE DELT2 STARTS OSCILLATING
   11 IER=1
   12 J=IEND
      GOTO 8
C
C     THERE ARE TWO IDENTICAL ARGUMENT VALUES IN VECTOR ARG
   13 IER=3
      GOTO 12
      END
C.....................................................................
      FUNCTION T(X,TRANS)
C  FUNCTION T SETS UP THE CORRECT ARGUMENT FOR CALLING TRANS AND
C  RETURNS THE REAL PART OF THE COMPLEX EVALUATION.
      COMPLEX C
      COMMON/TPT/G,N1,E,T1
      CALL TRANS(CMPLX(G,X/T1),C)
      T=REAL(C)
      RETURN
      END
C.....................................................................
      SUBROUTINE PEAK(P,Y,D,D1)
C  PEAK COMPUTES THE PEAK OF THE 3 POINTS, Y-D,Y, AND Y+D1, WHEN IT IS
C  KNOWN THAT D AND D1 ARE OF OPPOSITE SIGN. A SIMPLE 2ND-ORDER
C  FORMULA IS USED.
      IF(D1-D.NE.0.) GOTO 5
      P=Y
      RETURN
    5 P=Y-(D1+D)**2*.125/(D1-D)
      RETURN
      END
C.....................................................................
      SUBROUTINE DELTA2(N,X,A)
C  DELTA2 APPLIES AITKEN'S DELTA SQUARED TRANSFORMATION TO THE N
C  ELEMENTS OF ARRAY X. THE RESULTING N-2 ELEMENTS ARE PLACED IN THE
```

```
C  ARRAY A.
      DIMENSION X(1),A(1)
      NM2=N-2
      DO 5 I=1,NM2
      XI1=X(I+1)
      XI2=X(I+2)
      D=X(I)-XI1-XI1+XI2
      IF(D.NE.0.) GOTO 4
      A(I)=XI1
      GOTO 5
    4 A(I)=XI2-(XI2-XI1)**2/D
    5 CONTINUE
      RETURN
      END
C.......................................................................
      FUNCTION FCT(N,TRANS,IER)
C  FUNCTION FCT COMPUTES THE BROMWICH SUBINTERVAL FROM 2*PI*(N-1) TO
C  2*PI*N BY A HIGH-ORDER GAUSSIAN QUADRATURE.  THE VALUE IS CHECKED BY
C  THE NEXT HIGHEST-ORDER SUPPLIED. PROVISION IS MADE TO USE A LOWER
C  PAIR OF QUADRATURE FORMULAS IF THE ERROR IS TOO SMALL, AND A HIGHER
C  PAIR IF THE ERROR IS TOO LARGE.  THE ARGUMENT LIST IS AS FOLLOWS.
C           N - THE NUMBER OF THE INTERVAL.
C      TRANS - POINTS FOR DEFINITION.
C        IER - ERROR CODE
C              IER = 0 IF FCT IS SATISFACTORY.
C                  = -N IF THE ERROR CRITERION WAS NOT SATISFIED WITH
C                        THE 80 AND 96 POINT GAUSSIAN QUADRATURE
C                        FORMULAS.
      EXTERNAL TRANS
      COMMON/TPT/G,N1,EPS,T1
      COMMON/Q/PI,PI2,X,P
      DIMENSION Y(2)
C  P IS THE MIDPOINT OF THE INTERVAL.
      P=PI2*FLOAT(N)-PI
C  N2 CONTROLS THE ORDER OF THE GAUSSIAN FORMULA SELECTED, N1 IS
C  INITIALLY SET TO 5 IN TPOINT
      N2=N1
C  K IS THE FLAG TO INDICATE IF THE FIRST OR SECOND FORMULA IS CHOSEN.
      K=1
C  N1+1=11 MEANS FAILURE TO SATISFY THE ERROR CRITERION WITH AN 80-
C  AND 96-POINT GAUSSIAN QUADRATURE FORMULA.  RETURN IS MADE TO TPOINT
C  WHERE GC IS INCREASED TO GET FURTHER AWAY FROM SINGULARITIES.
   13 NP1=N1+1
      IF(NP1.LT.11) GOTO 15
      IER=-N
      N1=9
      RETURN
C  SELECT THE APPROPRIATE QUADRATURE FORMULA.
   15 GOTO(1,2,3,4,5,6,7,8,9,10), N2
    1 CALL  Q6(  Y(K),TRANS)
```

```
      GOTO 16
    2 CALL Q12(  Y(K),TRANS)
      GOTO 16
    3 CALL Q16(  Y(K),TRANS)
      GOTO 16
    4 CALL Q24(  Y(K),TRANS)
      GOTO 16
    5 CALL Q32(  Y(K),TRANS)
      GOTO 16
    6 CALL Q40(  Y(K),TRANS)
      GOTO 16
    7 CALL Q48(  Y(K),TRANS)
      GOTO 16
    8 CALL Q64(  Y(K),TRANS)
      GOTO 16
    9 CALL Q80(  Y(K),TRANS)
      GOTO 16
   10 CALL Q96(  Y(K),TRANS)
   16 IF(K.EQ.1) GOTO 20
C  CHECK TO SEE IF ERROR CRITERION IS SATISFIED.
      IF(ABS(Y(1)-Y(2)).LE.EPS*.1 ) GOTO 18
C  ERROR CRITERION IS NOT SATISFIED. INCREASE N1 AND N2 AND TRY AGAIN.
      Y(1)=Y(2)
      N1=N1+1
      N2=N1+1
      GOTO 13
C  ERROR CRITERION IS SATISFIED.
   18 FCT=Y(2)
      IER=0
C  CHECK TO SEE IF ERROR IS TOO SMALL.
      IF(ABS(Y(1)-Y(2)).GT.EPS*1.E-3) RETURN
C  ERROR IS TOO SMALL. DECREASE N1 FOR NEXT INTERVAL.
      N1=N1-1
      IF(N1.LT.1) N1=1
      RETURN
   20 K=2
      N2=NP1
      GOTO 15
      END
C.....................................................................
      SUBROUTINE  Q6(  Y,TRANS)
C  SUBROUTINE  Q6 COMPUTES A  6-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y.THE ZEROS OF THE LEGENDRE
C  POLYNOMIAL OF DEGREE 6 HAVE BEEN MULTIPLIED BY PI AND STORED IN X,
C  AND THE WEIGHTS HAVE BEEN MULTIPLIED BY -COS X AND STORED IN W
C  (BOTH IN DATA STATEMENTS).
      COMMON/C/PI,PI2,Z,P
      EXTERNAL TRANS
      DIMENSION X( 3),W( 3)
      DATA          X/2.929439   ,2.077251   ,.7496443  /,          W
```

```
     * /.1674834  ,.1749981 ,-.3424809 /
       Y=0.
       DO 5 I=1,3
       Z=X(I)
     5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
       RETURN
       END
C.....................................................................
       SUBROUTINE Q12(  Y,TRANS)
C  SUBROUTINE Q12 COMPUTES A 12-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
       COMMON/Q/PI,PI2,Z,P
       EXTERNAL TRANS
       DIMENSION X( 6),W( 6)
       DATA              X/3.083664 ,2.840368 ,2.418721 ,1.845114 ,
     * 1.155577 ,.3934324 /,             W/.4709620 E-1,.1021243 ,
     * .1200442 ,.5503602 E-1,-.9418876 E-1,-.2301119 /
       Y=0.
       DO 5 I=1,6
       Z=X(I)
     5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
       RETURN
       END
C.....................................................................
       SUBROUTINE Q16(  Y,TRANS)
C  SUBROUTINE Q16 COMPUTES A 16-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
       EXTERNAL TRANS
       COMMON/Q/PI,PI2,Z,P
       DIMENSION X( 8),W( 8)
       DATA              X/3.108295 ,2.96747,2.719461 ,2.373173 ,
     * 1.941115 ,1.438902 ,.8846836 ,.2984906/,           W/.2713741 E-1
     *,.6131218 E-1,.9680526 E-1,.8960946E-1,.5414073 E-1,-.2224613 E-1,
     * -.1156855 ,-.1810734 /
       Y=0.
       DO 5 I=1,8
       Z=X(I)
     5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
       RETURN
       END
C.....................................................................
       SUBROUTINE Q24(  Y,TRANS)
C  SUBROUTINE Q24 COMPUTES A 24-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
       EXTERNAL TRANS
       COMMON/Q/PI,PI2,Z,P
       DIMENSION X(12),W(12)
       DATA              X/3.126473 ,3.062200 ,2.947676 ,2.784757 ,
     *2.576112 ,2.325169 ,2.036046 ,1.713492 ,1.362802 ,.9897358 ,
     *.6004176 ,.2012407 /,             W/1.233982 E-2,2.844152 E-2,
     *4.344755 E-2,5.556317E-2,6.192873 E-2,5.902573 E-2,4.379624 E-2,
     *1.527986 E-2,-2.385162 E-2,-5.678623E-2,-.1038285 ,-.1253563 /
      Y=0.
      DO 5 I=1,12
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
C..................................................................
      SUBROUTINE Q32(  Y,TRANS)
C  SUBROUTINE Q32 COMPUTES A 32-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
      EXTERNAL TRANS
      COMMON/Q/PI,PI2,Z,P
      DIMENSION X(16),W(16)
      DATA           X/3.132997 ,3.096390 ,3.03089,2.937094 ,2.815876,
     *2.668367 ,2.495944 ,2.300218,2.083015,1.846364 ,1.592473,1.323714
     *,1.042596,.7517434 ,.4538721 ,.1517630 /,          W/7.018351 E-
     *3,1.625777 E-2,2.523663 E-2,3.355970 E-2,4.058366 E-2,4.539352 E-2
     *,4.687157 E-2,4.386647 E-2,3.545757 E-2,2.127599 E-2,1.805786 E-3,
     *-2.143759 E-2,-4.594979 E-2,-6.555327 E-2,-8.555589 E-2,-9.543046
     *E-2/
      Y=0.
      DO 5 I=1,16
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
      SUBROUTINE Q40(  Y,TRANS)
C..................................................................
C  SUBROUTINE Q40 COMPUTES A 40-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
      EXTERNAL TRANS
      COMMON/C/PI,PI2,Z,P
      DIMENSION X(20),W(20)
      DATA           X/3.136056 ,3.112458 ,3.070153 ,3.009384 ,
     * 2.930518 ,2.834027,2.720492,2.590596 ,2.445119 ,2.284938 ,
     * 2.111014 ,1.924395 ,1.726202 ,1.517627 ,1.299926 ,1.074406 ,
     * .8424249 ,.6053773,.3646889 ,.1218071 /,            W/4.521208 E-
     *3,1.049383 E-2,1.637917 E-2,2.205171 E-2,2.731698 E-2,3.189002 E-2
     *,3.539414 E-2,3.737815 E-2,3.735501E-2,3.486258E-2,2.954259 E-2,
     *2.122887 E-2,1.003042 E-2,-3.609110 E-3,-1.889359 E-2,-3.471256 E-
     *2,-4.973986 E-2,-6.258459 E-2,-7.197328 E-2,-7.693168 E-2/
      Y=0.
      DO 5 I=1,20
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
C..................................................................
```

```
      SUBROUTINE Q48(  Y,TRANS)
C  SUBROUTINE Q48 COMPUTES A 48-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
        EXTERNAL TRANS
      COMMON/Q/PI,PI2,Z,P
      DIMENSION X(24),W(24)
      DATA                X/3.137732 ,3.121267 ,3.091719 ,3.049203 ,
     *2.993899 ,2.926038 ,2.845903 ,2.753832 ,2.650211 ,2.535473 ,
     * 2.410101 ,2.274620 ,2.129599 ,1.975645 ,1.813405 ,1.643559 ,
     * 1.466819 ,1.283926 ,1.095549 ,.9027759 ,.7061163 ,.5064950 ,
     * .3047493 ,.1017253 /,               W/3.153323 E-3,7.326040 E-3,
     *1.146296 E-2,1.551287 E-2,1.940260 E-2,2.302528 E-2,2.623624E-2,
     *2.885332 E-2,3.066244 E-2,3.142922 E-2,3.091698 E-2,2.891059 E-2,
     *2.524498 E-2,1.983541 E-2,1.270611 E-2,4.012902 E-3,-5.944829 E-3,
     *-1.672663 E-2,-2.777043 E-2,-3.842928 E-2,-4.802281 E-2,
     * -5.589856 E-2,-6.149571 E-2,-6.440303 E-2/
      Y=0.
      DO 5 I=1,24
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
C.......................................................................
      SUBROUTINE Q64(  Y,TRANS)
C  SUBROUTINE Q64 COMPUTES A 64-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
      EXTERNAL TRANS
      COMMON/Q/PI,PI2,Z,P
      DIMENSION X(32),W(32)
      DATA     - -        X/3.139409 ,3.130095 ,3.113360 ,3.089242,
     * 3.057796 ,3.019098 ,2.973239,2.920328 ,2.860490 ,2.793867 ,
     * 2.720617 ,2.640915 ,2.554948 ,2.462922 ,2.365053 ,2.261576 ,
     * 2.152733 ,2.038785 ,1.920001 ,1.796663 ,1.669064 ,1.537505 ,
     * 1.402300 ,1.263769,1.122240 ,.9780496 ,.8315392 ,.6830565 ,
     * .5329537 ,.3815857,.2293147 ,7.649870 E-2/
      DATA               W/1.783276 E-3,4.146759 E-3,6.501866 E-3,
     *8.834640 E-3,1.112895 E-2,1.336217 E-2,1.550370 E-2,1.751406 E-2,
     *1.934453 E-2,2.093731 E-2,2.222649 E-2,2.313983E-2,2.360136 E-2,
     *2.353489 E-2,2.286831 E-2,2.153855 E-2,1.949706E-2,1.671528 E-2,
     *1.318996 E-2,8.947691 E-3,4.048239 E-3,-1.413725 E-3,-7.308989E-3,
     *-1.347644 E-2,-1.972812 E-2,-2.585660 E-2,-3.164430 E-2,-3.687439
     *E-2,-4.134236E-2,-4.486756 E-2,-4.730388 E-2,-4.854856 E-2/
      Y=0.
      DO 5 I=1,32
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
C.......................................................................
      SUBROUTINE Q80(  Y,TRANS)
```

```
C  SUBROUTINE Q80 COMPUTES A 80-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
      EXTERNAL TRANS
      COMMON/Q/PI,PI2,Z,P
      DIMENSION X(40),W(40)
      DATA    - - ;       X/3.140191 ,3.134209 ,3.123458 ,3.107950 ,
     1 3.087710 ,3.062767 ,3.033161 ,2.998935 ,2.960143 ,2.916843 ,
     2 2.869101 ,2.816989 ,2.760588 ,2.699983 ,2.635267 ,2.566537 ,
     3 2.493899 ,2.417463 ,2.337346 ,2.253669 ,2.166561 ,2.076153 ,
     4 1.982583 ,1.885994 ,1.786533 ,1.684352 ,1.579606 ,1.472454 ,
     5 1.363060 ,1.251590 ,1.138214 ,1.023105 ,.9064376 ,.7883901 ,
     6 .6691420 ,.5488749 ,.4277719 ,.3060175 ,.1837971 ,6.129682 E-2/
      DATA                W/1.144949 E-3,2.663461E-3,4.179626 E-3,5.687702
     1E-3,7.182466 E-3,8.656981 E-3,1.010209 E-2,1.150603 E-2,1.285421 E
     2-2,1.412899 E-2,1.530967 E-2,1.637252 E-2,1.729112 E-2,1.803665E-2
     3,1.857859 E-2,1.888542 E-2,1.892564 E-2,1.866888 E-2,1.808719 E-2,
     41.715643 E-2,1.585776 E-2,1.417906 E-2,1.211641 E-2,9.675300 E-3,6
     5.871675 E-3,3.732713 E-3,2.971648 E-4,-3.384710 E-3,-7.251709 E-3,
     6-1.123276 E-2,-1.524848 E-2,-1.921330 E-2,-2.303786 E-2,
     7 -2.663188 E-2,-2.990720 E-2,-3.278083 E-2,-3.517804 E-2,
     8 -3.703520 E-2,-3.830221 E-2,-3.894454 E-2/
      Y=0.
      DO 5 I=1,40
      Z=X(I)
    5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
      RETURN
      END
C..........................................................................
      SUBROUTINE Q96(  Y,TRANS)
C  SUBROUTINE Q96 COMPUTES A 96-POINT GAUSSIAN QUADRATURE OVER THE
C  INTERVAL AND RETURNS THE ANSWER Y. SEE Q6 FOR THE MEANING OF X AND W
      EXTERNAL TRANS
      COMMON/Q/PI,PI2,Z,P
      DIMENSION X(48),W(48)
      DATA                X/3.140617 ,3.136454 ,3.128969 ,3.118169 ,
     1 3.104064 ,3.086669 ,3.066003 ,3.042088 ,3.014950 ,2.984616 ,
     2 2.951119 ,2.914495 ,2.874782 ,2.832022 ,2.786261 ,2.737548 ,
     3 2.685933 ,2.631472 ,2.574222 ,2.514244 ,2.451602 ,2.386361 ,
     4 2.318592 ,2.248365 ,2.175756 ,2.100841 ,2.023699 ,1.944413 ,
     5 1.863066 ,1.779745 ,1.694538 ,1.607535 ,1.518828 ,1.428512 ,
     6 1.336682 ,1.243435 ,1.148871 ,1.053089 ,.9561907 ,.8582794 ,
     7 .7594585 ,.6598328 ,.5595079 ,.4585899 ,.3571860 ,.2554036 ,
     8 .1533505 ,5.113490 E-2/
      DATA                W/7.967917 E-4,1.853936 E-3,2.910500 E-3,3.963467
     * E-3,5.010672 E-3,6.049410 E-3,7.076207 E-3,8.086678 E-3,9.075404
     *E-3,1.003584 E-2,1.096023 E-2,1.183960 E-2,1.266371 E-2,1.342112 E
     *-2,1.409927 E-2,1.468459 E-2,1.516267 E-2,1.551853 E-2,1.573687 E-
     *2,1.580239 E-2,1.570022 E-2,1.541629 E-2,1.493784 E-2,1.425388 E-2
     *,1.335568 E-2,1.223728 E-2,1.089595 E-2,9.332676 E-3,7.552470 E-3,
     *5.564738 E-3,3.383468 E-3,1.027341 E-3,-1.480296 E-3,-4.111580 E-3
```

```
 *,-6.834439 E-3,-9.613038 E-3,-1.240836 E-2,-1.517894 E-2,-1.788169
 * E-2,-2.047281 E-2,-2.290887 E-2,-2.514773 E-2,-2.714973 E-2,-2.88
 *7858 E-2,-3.030242 E-2,-3.139462 E-2,-3.213454 E-2,-3.250807 E-2/
  Y=0.
  DO 5 I=1,48
  Z=X(I)
5 Y=Y+(T(P+Z,TRANS) + T(P-Z,TRANS))*W(I)
  RETURN
  END
```